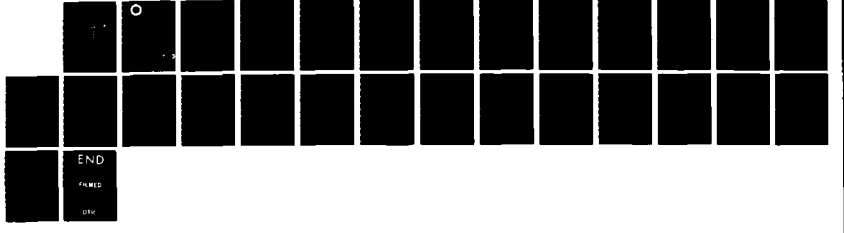
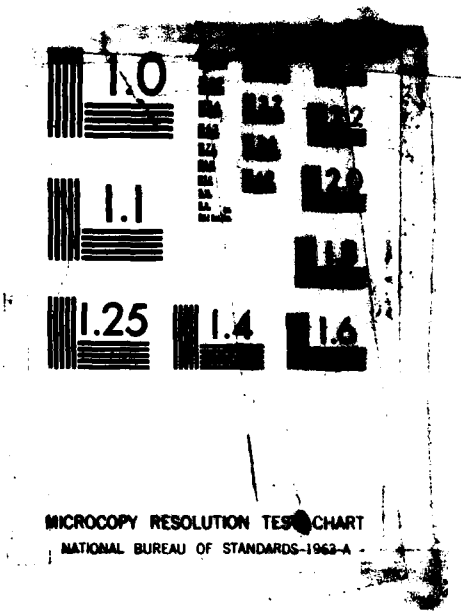
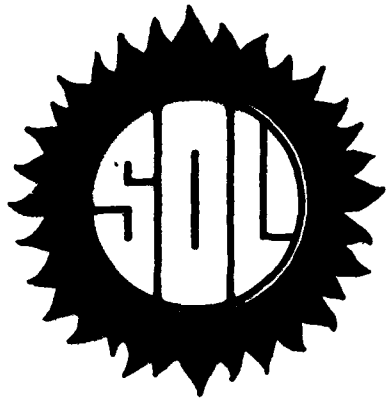
AD-A158 212 ON PROJECTED NEWTON BARRIER METHODS FOR LINEAR PROGRAMMING AND AN EQUIVAL.. (U) STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB P E GILL ET AL. JUL 85 1/1

UNCLASSIFIED SOL-85-11 ARO-21592.6-MA N00014-85-K-0343 F/G 12/1 NL




1.0
1.1
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



Systems
Optimization
Laboratory

ON PROJECTED NEWTON BARRIER METHODS FOR LINEAR PROGRAMMING AND AN EQUIVALENCE TO KARMARKAR'S PROJECTIVE METHOD

by

Philip E. Gill, Walter Murray,
Michael A. Saunders, J.A. Tomlin† and Margaret H. Wright

TECHNICAL REPORT SOL 85-11

July 1985

Department of Operations Research
Stanford University
Stanford, CA 94305

2

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

ON PROJECTED NEWTON BARRIER METHODS FOR LINEAR PROGRAMMING AND AN EQUIVALENCE TO KARMARKAR'S PROJECTIVE METHOD

by

Philip E. Gill, Walter Murray,
Michael A. Saunders, J.A. Tomlin† and Margaret H. Wright

TECHNICAL REPORT SOL 85-11

July 1985

†Ketron Incorporated, 201 San Antonio Circle, Mountain View, CA 94040.

Research and reproduction of this report were partially supported by the National Science Foundation Grants DCR-8413211 and ECS-8312142; U.S. Department of Energy Contract DE-AM03-76SF00326, PA# DE-AT03-76ER72018; Office of Naval Research Contract N00014-85-K-0343; and U.S. Army Research Office Contract DAAG29-84-K-0156.

(A)

On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method

Philip E. Gill†, Walter Murray†,
Michael A. Saunders†, J. A. Tomlin‡ and Margaret H. Wright†

ABSTRACT

(1) The authors
~~We~~ discuss interior-point methods for linear programming derived by applying a logarithmic barrier transformation and performing projected Newton steps for a sequence of barrier parameters. Under certain conditions, one of these "projected Newton barrier" methods is shown to be equivalent to Karmarkar's (1984) projective method for linear programming.

Details are given of a specific barrier algorithm and its practical implementation. Numerical results are given for several nontrivial test problems. Additional keywords: tables (data); numerical analysis; iterations; computations; least squares method. (2)

The research of the Stanford authors was supported by the U.S. Department of Energy Contract DE-AM03-76SF00326, PA No. DE-AT03-76ER72018; National Science Foundation Grants DCR-8413211 and ECS-8312142; the Office of Naval Research Contract N00014-85-K-0343; and the U.S. Army Research Office Contract DAAG29-84-K-0156.

† Systems Optimisation Laboratory, Department of Operations Research, Stanford University, Stanford, California 94305.

‡ Ketron Incorporated, 201 San Antonio Circle, Mountain View, California 94040.

1. Introduction

Interest in linear programming methods arises in at least two different contexts: theoretical and practical. The long-established simplex method, developed by G. B. Dantzig in the late 1940's, has been known from the beginning to be of combinatorial complexity in the worst case. However, in practice it tends to require a number of iterations that is approximately *linear* in the problem dimension. Furthermore, a typical iteration of the simplex method involves relatively little work. After computing an initial factorization of a square matrix (the *basis*), each subsequent simplex iteration requires only a rank-one update of this square matrix. Although linear programs are often very large, the constraint matrix is normally very *sparse*. Sparse-matrix techniques have developed to the point where the factorization and updates required in the simplex method can be performed not only rapidly, but also with assured numerical stability (see the survey by Gill *et al.*, 1984). From a practical viewpoint, these two features — a typically linear number of iterations, and fast methods for performing each iteration — imply that the simplex method is an effective and reliable algorithm for linear programming, despite its seemingly unfavorable complexity.

Many researchers, beginning with Dantzig himself, have observed the seemingly unsatisfactory feature that the simplex method traverses the boundary of the feasible region. From the outset, attempts have been made to develop practical linear programming methods that cross the *interior* of the feasible region — for example, von Neumann (1947), Hoffman *et al.* (1953), Tompkins (1955, 1957) and Frisch (1957). Such methods have sometimes involved the application of *nonlinear* techniques to linear programs. However, none of these methods has previously been claimed, even by its developers, to be competitive in speed with the simplex method for general linear programs.

On the theoretical side, researchers attempted for many years to develop a linear programming algorithm with only polynomial complexity. In 1979, to the accompaniment of wide publicity, this issue was resolved when Khachiyan (1979) presented a worst-case polynomial-time method based on a nonlinear geometry of shrinking ellipsoids. Although initially it was thought that the ellipsoid methods might be as fast in practice as the simplex method, these hopes have not been realized. Broadly speaking, there are two major difficulties: first, the number of iterations tends to be very large; second, the computation associated with each iteration is much more costly than a simplex iteration because sparse-matrix techniques are not applicable.

Within the past year, interest in linear programming has been intensified by the publication (Karmarkar, 1984) and discussion of a linear programming algorithm that is not only polynomial in complexity, but also is claimed to be *much faster* than the simplex method for practical problems.

In Section 2, we first examine the well known barrier-function approach to solving optimization problems with inequality constraints, and derive a representation for the Newton search direction associated with the subproblem. In Section 3, we show a formal equivalence between the Newton search direction and the direction associated with Karmarkar's (1984) algorithm. Section 4 describes a complete interior-point method for linear programming based on the barrier

transformation, and Section 5 gives some numerical results obtained with a preliminary implementation of that method. The implications of these results and directions for future research are discussed in Section 6.

1.2. Notation. The term *projective method* will denote the algorithm given by Karmarkar (1984) for the linear program (3.1); see below. The term *barrier method* will often be used as an abbreviation for *projected Newton barrier method*. The vector norm $\|\cdot\|$ will always denote the Euclidean norm $\|v\|_2 = (v^T v)^{1/2}$.

2. A Barrier-Function Approach

2.1. Applying a barrier transformation to a linear program. Barrier-function methods treat *inequality* constraints by creating a *barrier function*, which is a combination of the original objective function and a weighted sum of functions with a positive singularity at the constraint boundary. (Many barrier functions have been proposed; we consider only the logarithmic barrier function, first suggested by Frisch, 1955.) As the weight assigned to the singularities approaches zero, the minimum of the barrier function approaches the minimum of the original constrained problem. Barrier-function methods require a strictly feasible starting point for each minimization, and generate a sequence of strictly feasible iterates. (The definitive work on barrier functions remains Fiacco and McCormick, 1968; overviews are given by Fletcher, 1981, and Gill, Murray and Wright, 1981.)

Consider applying a barrier-function method to the following linear program:

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} && c^T x \\ &\text{subject to} && Ax = b, \qquad x \geq 0, \end{aligned} \tag{2.1}$$

where A is an $m \times n$ matrix with $m \leq n$. The subproblem to be solved within a barrier-function method is:

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} && F(x) \equiv c^T x - \mu \sum_{j=1}^{n} \ln x_j \\ &\text{subject to} && Ax = b, \end{aligned} \tag{2.2}$$

where the scalar μ ($\mu > 0$) is known as the *barrier parameter* and is specified for each subproblem. The equality constraints cannot be treated by a barrier transformation, and thus are handled directly. If $x^*(\mu)$ is the solution of (2.2), then $x^*(\mu) \to x^*$ as $\mu \to 0$, where x^* is a solution of (2.1) (see, e.g., Fiacco and McCormick, 1968). Very strong order relations can be derived concerning $x^*(\mu)$ and $c^T x^*(\mu)$ (see, e.g., Mifflin, 1972a, b; Jittorntrum, 1978; Jittorntrum and Osborne, 1978). In particular, when (2.1) is *nondegenerate*,

$$\|x^*(\mu) - x^*\| = O(\mu) \tag{2.3a}$$

for sufficiently small μ. When (2.1) is degenerate, the corresponding relation is

$$\|x^*(\mu) - x^*\| = O(\sqrt{\mu}). \tag{2.3b}$$

2.2. Solution of the subproblem. Given a linearly constrained problem of the form

$$\text{minimize } F(x) \quad \text{subject to} \quad Ax = b, \tag{2.4}$$

a standard approach is to use a *feasible-point descent method* (see, e.g., Gill, Murray and Wright, 1981). The current iterate x always satisfies $Ax = b$, and the next iterate $\bar{x}$ is defined as

$$\bar{x} = x + \alpha p, \tag{2.5}$$

where p is an n-vector (the *search direction*), and α is a positive scalar (the *steplength*). The computation of p and α must ensure that $A\bar{x} = b$ and $F(\bar{x}) < F(x)$.

The *Newton search direction* associated with (2.4) is defined as the step to the minimum of the quadratic approximation to $F(x)$ derived from the local Taylor series, subject to retaining feasibility. Thus, the Newton search direction is the solution of the following quadratic program:

$$\begin{aligned} &\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad g^T p + \frac{1}{2} p^T H p \\ &\text{subject to} \quad Ap = 0, \end{aligned} \tag{2.6}$$

where $g \equiv \nabla F(x)$ and $H \equiv \nabla^2 F(x)$. If π is the vector of Lagrange multipliers for the constraints in (2.6), then the required solution satisfies the linear system

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -p \\ \pi \end{pmatrix} = \begin{pmatrix} g \\ 0 \end{pmatrix}. \tag{2.7}$$

Note that π converges to the Lagrange multipliers for the constraints $Ax = b$ in the original problem (2.4).

2.3. The projected Newton search direction. When $F(x)$ is the barrier function in (2.2), its derivatives are

$$g(x) = c - \mu D^{-1} e \qquad \text{and} \qquad H(x) = \mu D^{-2},$$

where

$$D = \text{diag}\,(x_j), \quad j = 1, \ldots, n, \tag{2.8}$$

and $e = (1, 1, \ldots, 1)^T$. Note that g and H are well defined only if $x_j \neq 0$ for all j.

Let p_B (the *projected Newton barrier direction*) denote the Newton search direction defined by (2.6) when F is the barrier function of (2.2). The associated Lagrange multipliers will be denoted by π_B. Since $H(x)$ is positive definite when $x > 0$, p_B is finite and unique, and is a *descent direction* for $F(x)$, i.e., $(c - \mu D^{-1} e)^T p_B < 0$.

It follows from (2.7) that p_B and π_B satisfy the equation

$$\begin{pmatrix} \mu D^{-2} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -p_B \\ \pi_B \end{pmatrix} = \begin{pmatrix} c - \mu D^{-1} e \\ 0 \end{pmatrix}. \tag{2.9}$$

Rewriting (2.9) in terms of a vector r_π defined by $Dr_\pi = -\mu p_\pi$, we see that r_π and π_π satisfy

$$\begin{pmatrix} I & DA^T \\ AD & 0 \end{pmatrix} \begin{pmatrix} r_\pi \\ \pi_\pi \end{pmatrix} = \begin{pmatrix} Dc - \mu e \\ 0 \end{pmatrix}. \tag{2.10}$$

It follows that π_π is the solution and r_π the optimal residual of the following linear least-squares problem:

$$\underset{\pi}{\text{minimize}} \ \|Dc - \mu e - DA^T\pi\|. \tag{2.11}$$

The projected Newton barrier direction is then

$$p_\pi = -(1/\mu)Dr_\pi. \tag{2.12}$$

For a given positive μ, Newton's method will eventually reach a domain in which unit steps along the directions p_π will be feasible. The iterates can then be expected to converge quadratically. In general, the smaller μ, the smaller the attractive domain. The algorithm remains well defined as μ tends to zero, and the limiting case can be safely simulated (in practice) by using a very small value such as $\mu = 10^{-15}$.

Note that feasible-direction methods are independent of the scaling of the search direction, if the steplength algorithm is chosen appropriately. We could therefore define the barrier search direction to be

$$p_\pi = -Dr_\pi \tag{2.13}$$

for any $\mu \geq 0$. The ideal Newton step would then be $\alpha = 1/\mu$.

The barrier search direction (2.13) with $\mu = 0$ is used in an algorithm proposed by Vanderbei, Meketon and Freedman (1985). We see that such an algorithm has no domain of quadratic convergence.

2.4. Upper bounds. The barrier transformation and the associated Newton search direction can also be defined for linear programs with upper and lower bounds on the variables, of the form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b, \quad \ell \leq x \leq u. \end{aligned}$$

The subproblem analogous to (2.2) is

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x - \mu \sum_{j=1}^{n} \ln(x_j - \ell_j) - \mu \sum_{j=1}^{n} \ln(u_j - x_j) \\ \text{subject to} \quad & Ax = b. \end{aligned}$$

The Hessian of the associated barrier function will be positive definite only if at least one of ℓ_j or u_j is finite for every j. In this case, the least-squares problem analogous to (2.11) is

$$\underset{\pi}{\text{minimize}} \ \|Dc - \mu De - DA^T\pi\|.$$

Here, the matrices D and $\bar{D}$ are defined by $D = \mathrm{diag}(\delta_j)$ and $\bar{D} = \mathrm{diag}(\bar{\delta}_j)$, where

$$\delta_j = 1/(1/s_j^2 + 1/t_j^2)^{\frac{1}{2}} \quad \text{and} \quad \bar{\delta}_j = \delta_j(1/s_j - 1/t_j),$$

with $s_j \quad x_j - \ell_j$ and $t_j = u_j - x_j$. For simplicity, the remainder of the discussion will assume that the bounds are of the form in (2.1), i.e., $0 \le x_j \le \infty$, except for the artificial variable discussed in Section 4.2.

3. Relationship with Karmarkar's Projective Method

In this section, we show the connection between the barrier and projective methods. We assume that the reader is familiar with the projective method; a good description is given in Todd and Burrell (1985).

3.1. Summary of the projective method. In the projective method, the linear program is assumed to be of the special form

$$\begin{aligned} &\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad c^T x \\ &\text{subject to} \quad Cx = 0, \quad e^T x = 1, \quad x \ge 0. \end{aligned} \tag{3.1}$$

Let x_K^* denote a solution of (3.1). It is also assumed that

$$c^T x_K^* = 0, \tag{3.2}$$

and that $Ce = 0$. (These assumptions can always be assured by transforming the problem.)

The optimality conditions for (3.1) imply that

$$c = C^T \lambda_C + e\lambda_e + \eta, \tag{3.3}$$

where η is the Lagrange multiplier vector for the bound constraints of (3.1). The complementarity conditions at x_K^* imply that

$$x_j^* \eta_j = 0, \quad j = 1, \ldots, n, \tag{3.4}$$

where x_j^* denotes the j-th component of x_K^*. Taking the inner product of e and x_K^* and using (3.2)-(3.4), we obtain $\lambda_e \, e^T x_K^* = 0$. Since $e^T x_K^* = 1$, it follows that $\lambda_e = 0$.

Any strictly positive diagonal matrix D defines the following projective transformations, which relate any strictly feasible point x and the transformed point x':

$$x' = \frac{1}{e^T D^{-1} x} D^{-1} x, \qquad x = \frac{1}{e^T D x'} D x'. \tag{3.5}$$

In the projective method, given an iterate x, D is defined as $\mathrm{diag}(x_1, \ldots, x_n)$. (Note that D is the same as the diagonal matrix (2.8) associated with the barrier method, and that $De = x$.) The next iterate in the *transformed* space is given by

$$x' = x' - \alpha' r_K, \tag{3.6}$$

where $r_K = Dc - DC^T\pi_K - \phi e$ is the optimal residual of the linear least-squares problem

$$\underset{\pi,\phi}{\text{minimize}} \ \|Dc - (DC^T \quad e)\begin{pmatrix}\pi\\ \phi\end{pmatrix}\|. \tag{3.7}$$

The steplength α' in (3.6) is chosen to ensure strict feasibility of x' as well as descent in the transformed "potential function" (see Karmarkar, 1984, for details).

The new iterate z_K in the original parameter space is obtained by applying the transformation (3.5) to x', so that

$$z_K = \frac{1}{c^TD(x' - \alpha' r_K)} D(x' - \alpha' r_K) = \gamma(x - \tilde{\alpha} D r_K),$$

where γ is chosen to make $c^T z_K = 1$.

3.2. Properties of the projective method. Let π_C be defined as the solution of the least-squares problem

$$\underset{\pi}{\text{minimize}} \ \|Dc - DC^T\pi\|,$$

and let

$$\eta_C = c - C^T\pi_C,$$
$$\mu_C = x^TD\eta_C.$$

For reference, we state some properties of various quantities appearing in the projective method.

Lemma 3.1. *The solution of (3.7) is* $\pi_K = \pi_C$ *and* $\phi = c^Tx/n$.

Lemma 3.2. *In the projective method,*

$$r_K = D\eta_C - \phi e,$$
$$\tilde{\alpha} = n\alpha',$$
$$\gamma = 1/\big(1 + \tilde{\alpha}(\phi - \mu_C)\big),$$

and the new iterate may be written as

$$p_K = \mu_C x - D^2\eta_C,$$
$$z_K = x + \tilde{\alpha}\gamma p_K.$$

3.3. Relationship with the barrier search direction. When the barrier method is applied to problem (3.1), we obtain π_B and θ as the solution of the least-squares problem

$$\underset{\pi,\theta}{\text{minimize}} \ \|Dc - \mu e - D(C^T \quad e)\begin{pmatrix}\pi\\ \theta\end{pmatrix}\|, \tag{3.8}$$

and take

$$r_B = Dc - \mu e - DC^T\pi_B - \theta De,$$
$$p_B = -(1/\mu)Dr_B,$$
$$z_B = x + \alpha p_B,$$

for some μ and α. We assume that the matrices $(DC^T \quad e)$ and $(C^T \quad e)$ have full rank, so that the solutions of (3.7) and (3.8) are unique.

Lemma 3.3. *If the barrier parameter is chosen to be $\mu = \mu_C$, then $\theta = 0$ and $\pi_B = \pi_C$.*

Lemma 3.4. *If $\mu = \mu_C$, then*

$$r_B = D\eta_C - \mu_C e,$$
$$p_B = (1/\mu_C)(\mu_C x - D^2\eta_C),$$
$$x_B = x + \alpha p_B.$$

Comparing Lemmas 3.2 and 3.4, the main result now follows.

Theorem 3.1. *Suppose the projective method and the barrier method are applied to problem (3.1). If the barrier parameter is $\mu = \mu_C$, the search directions p_B and p_K are parallel. Further, if the steplengths satisfy $\alpha = \tilde{\alpha}\gamma\mu_C$, the iterates x_B and x_K are identical.*

Theorem 3.1 is an existence result, showing that a special case of the barrier method would follow the same path as the projective method. This does not mean that the barrier method *should* be specialized. For example, the value $\mu = \mu_C$ is admissible in the barrier method only if it is positive. As it happens, μ_C tends to zero and is likely to be positive in a neighborhood of the solution. It could therefore be a satisfactory choice as the solution is approached.

Similarly, whatever the choice of μ, as the barrier method converges to a solution of the original problem, θ must converge to λ_e, which is zero. This is consistent with the choice $\mu = \mu_C$, which gives $\theta = 0$ directly.

4. A Projected Newton Barrier Algorithm

In this section, we give details of the barrier algorithm used to obtain the numerical results of Section 5. Each iteration is of the form $\bar{x} = x + \alpha p$ (2.5), where the search direction is the barrier direction p_B defined by (2.11)–(2.12), and *the barrier parameter may be changed at every iteration.* Any tolerances described are intended to be suitable for machines whose relative precision is about 10^{-15}.

Alternative approaches to certain parts of the algorithm are discussed in Section 6.

4.1. The main steps. At the start of each iteration, the quantities μ, x, π and η are known, where $\mu > 0$, $x > 0$, $Ax = b$, and $\eta = c - A^T\pi$. For computational reasons we compute a *correction* to π at each stage, since a good estimate is available from the previous iteration. The main steps of the iteration then take the following form.

1. Define $D = \text{diag}(x_j)$ and compute $r = D\eta - \mu e$.
2. Terminate if μ and $\|r\|$ are sufficiently small.
3. If appropriate, reduce μ and recompute $r = D\eta - \mu e$.
4. Solve the least-squares problem

$$\underset{\delta\pi}{\text{minimize}} \ \|r - DA^T\delta\pi\|. \tag{4.1}$$

5. Update $\pi \leftarrow \pi + \delta\pi$, $\eta \leftarrow \eta - A^T\delta\pi$, and set $r = D\eta - \mu e$, $p = -(1/\mu)Dr$.
6. Find α_M, the maximum step α such that $x + \alpha p \geq 0$.
7. Determine a steplength $\alpha \in (0, \alpha_M)$ at which the barrier function $F(x + \alpha p)$ is suitably less than $F(x)$.
8. Update $x \leftarrow x + \alpha p$.

All iterates satisfy $Ax = b$, $x > 0$, and the vectors π and η approximate the dual variables and reduced costs for the original linear program.

The vector r serves two purposes. In step 4, r is the right-hand side for the least-squares problem, and in step 5 it becomes the residual vector at the least-squares solution. In either case, $r \to 0$ as convergence occurs, so that $D\eta \to \mu e$. Hence, the reduced-cost estimates should ultimately satisfy $\eta > 0$, as one might expect.

4.2. The feasibility phase. In order to apply the barrier algorithm to (2.1), a strictly feasible starting point is necessary. Such a point may be found by the following "phase 1" procedure in which a barrier method is applied to a modified linear program. For any given initial point $x_0 > 0$, we define $\xi_0 s = b - Ax_0$ with $\|s\| = 1$, and solve the modified linear program

$$\begin{aligned} &\underset{x,\xi}{\text{minimize}} && \xi \\ &\text{subject to} && (A \quad s)\begin{pmatrix} x \\ \xi \end{pmatrix} = b, \quad x \geq 0, \quad \xi \geq 0, \end{aligned} \tag{4.2}$$

using the feasible starting point $x_0 > 0$, $\xi_0 = \|b - Ax_0\|$. (Note that, even if A is sparse, the additional column s in (4.2) will in general be *dense*.) In our experiments we have used $x_0 = \|b\|e$.

When $\xi = 0$, a suitable point has been found. Since the barrier transformation will not allow ξ to reach the desired value of zero, ξ must be treated differently from the other variables in solving (4.1) with a barrier algorithm. In our implementation, the search direction p and the maximum step α are computed as if the variable ξ were subject to the bound $\xi \geq -1$. If the step α causes ξ to become negative, an appropriate shorter step is taken and phase 1 is terminated. The original linear program is presumed to be infeasible if the final ξ is positive for a sufficiently small value of μ.

As an alternative, we note that the convergence of the barrier method appears to be moderately insensitive to the choice of linear objective function. This suggests a single-phase algorithm in which an objective function of the form $\omega c^T x + \xi$ is used in (4.2), for some positive value of the scalar ω. When ξ reaches zero, it can thereafter be excluded from the problem. If a single value of ω can be retained at every iteration, only a slight change in the definition of the linear program is required after a feasible point is found. Some preliminary results with ω fixed at $0.01/\|c\|$ seem promising; see Section 5. In general, a sequence of decreasing values of ω may be needed to ensure that a feasible point is always obtained if one exists.

4.3. Solution of the least-squares subproblems. For problems of even moderate size, the time required to perform an iteration will be dominated by solution of the least-squares problem (4.1). The widespread interest in interior-point methods has arisen because of their reported speed on large-scale linear programs. Consequently, problem (4.1) must be solved when A is large and sparse. Fortunately, methods for sparse least-squares problems have improved dramatically in the past decade. (For a recent survey, see Heath, 1984.)

An obvious approach to minimizing $\|r - DA^T\delta\pi\|$ is to solve the associated normal equations

$$AD^2A^T\delta\pi = ADr \tag{4.3}$$

using the Cholesky factorization $AD^2A^T = R^TR$ with R upper triangular. Reliable software exists for factorizing symmetric definite systems, notably *SPARSPAK-A* (George and Liu, 1981; George and Ng, 1984). If the original LP (2.1) is non-degenerate, the matrix AD^2A^T will be non-singular even at the solution. However, for a degenerate problem, AD^2A^T becomes increasingly ill-conditioned as the solution is approached, and the accuracy of the computed version of AD^2A^T correspondingly deteriorates. Furthermore, any dense columns in A (such as s in phase 1) degrade the sparsity of R.

To alleviate these difficulties, we have used a "hybrid" method in which the least-squares problems are solved by a conjugate-gradient method (*LSQR*; Paige and Saunders, 1982) with a triangular preconditioner R. Thus, an iterative method is applied to

$$\underset{z}{\text{minimize}} \ \|r - (DA^TR^{-1})z\|, \tag{4.4}$$

and the correction $\delta\pi$ is recovered by solving $R\,\delta\pi = z$.

The preconditioner R comes from the Cholesky factorization of a sparse matrix that approximates AD^2A^T. Thus,

$$AD^2A^T \approx \tilde{A}D^2\tilde{A}^T = R^TR, \tag{4.5}$$

where $\tilde{A}$ and R are obtained as follows.

1. Before beginning the barrier algorithm, a preliminary row permutation is obtained from the symbolic factorization of a matrix $\tilde{A}\tilde{A}^T$, where $\tilde{A}$ is A with certain columns replaced by zero. For the results of Section 5, we excluded the artificial vector s in (4.2) and any columns of A containing 50 or more nonzeros. Subroutines *Genqmd* and *Smbfct* of George and Liu (1981) were then used to obtain a minimum-degree ordering P and to set up appropriate data structures for the subsequent numerical factorizations.

2. At each iteration of the barrier algorithm, further columns and/or rows of $\tilde{A}$ may be replaced by zero: columns for which $x_j \le 10^{-6}$, and rows that have been marked for exclusion during earlier iterations. Subroutine *Gsfct* of George and Liu (1981) is then used to obtain the Cholesky factorization

$$P\tilde{A}D^2\tilde{A}^TP^T = U^TU, \qquad U \text{ upper triangular},$$

   with the proviso that if a diagonal element of U satisfies $u_{ii}^2 \le 10^{-12}$, then the i-th row of U is replaced by e_i^T, and the i-th row of $P\tilde{A}$ is marked for exclusion in later iterations. The preconditioner for (4.4) is then $R = UP$.

3. After each iteration, any variables satisfying $x_j \le 10^{-10}$ are changed to zero for the remaining iterations. This (conservative) test is unlikely to remove the "wrong" variables from the problem, but it allows some economy in computing R and solving the least-squares problems.

The performance of *LSQR* is strongly affected by the quality of the preconditioner, and by the specified convergence tolerance `ATOL` (see Paige and Saunders, 1982). With the present implementation, we have $AD^2A^T = R^TR + E_1 + E_2$, where E_1 has low rank and $\|E_2\|$ is small; the value of `ATOL` is taken as 10^{-12}. In this situation, *LSQR* typically requires only one or two iterations to achieve acceptable accuracy in phase 2, and only two or three iterations in phase 1.

There is scope in future work for degrading the approximation (4.5) to obtain a sparser R more quickly, at the expense of further iterations in *LSQR*. In fact, Gay (1985) has reported considerable success in the analogous task of preconditioning the symmetric conjugate-gradient method in order to solve the normal equations (4.3). We discuss this further in Section 6.1.

4.4. Determination of the steplength. The steplength α in (2.5) is intended to ensure a reduction in the barrier function $F(x)$ in (2.2) at every iteration. Let $f(\alpha)$ denote $F(x + \alpha p)$, treated as a function of α, and let α_M be the largest positive feasible step along p. When $p = p_B$, $f'(0) < 0$; by construction of a positive singularity at the boundary of the feasible region, $f'(\alpha_M) = +\infty$. Thus, there must exist a point α^* in the interval $(0, \alpha_M)$ such that

$f'(\alpha^*) = 0$. Because of the special form of f, α^* is unique and is the univariate minimizer of $f(\alpha)$ for $\alpha \in [0, \alpha_M]$.

In our algorithm, α is an approximation to a zero of the function $f'(\alpha)$. In order to obtain a "sufficient decrease" in F (in the sense of Ortega and Rheinboldt, 1970), an acceptable α is any member of the set

$$\Gamma = \{\alpha : |f'(\alpha)| \leq -\beta f'(0)\},$$

where β is a number satisfying $0 \leq \beta < 1$. (The smaller the value of β, the closer the approximation of α to a zero of f'.)

The computation of an acceptable steplength involves an iterative procedure for finding a zero of f'. Many efficient algorithms have been developed for finding the zero of a general univariate function (see, e.g., Brent, 1973), based on iterative approximation by a low-order polynomial. However, such methods tend to perform poorly in the presence of singularities. In order to overcome this difficulty, special steplength algorithms have been devised for the logarithmic barrier function (e.g., Fletcher and McCann, 1969; Murray and Wright, 1976). These special procedures are based on approximating $f(\alpha)$ by a function with a similar singularity.

Given an interval I such that $\alpha^* \in I$ and $\Gamma \subset I$, a new interval $\bar{I}$ ($\bar{I} \subset I$) is generated using α_ϕ, the zero of a simple monotonic function $\phi(\alpha)$ that approximates $f'(\alpha)$. Let $\alpha_B \in I$ be the current best estimate of α^*. Define the function $\phi(\alpha)$ to be

$$\phi(\alpha) = \gamma_1 + \frac{\gamma_2}{\alpha_M - \alpha},$$

where the coefficients γ_1 and γ_2 are chosen such that $\phi(\alpha_B) = f'(\alpha_B)$ and $\phi'(\alpha_B) = f''(\alpha_B)$. The new estimate of the zero of $f'(\alpha)$ is then given by

$$\alpha_\phi = \alpha_M + \gamma_2/\gamma_1.$$

Using this prescription, a *sequence* of intervals $\{I_j\}$ is generated such that $I_0 = [0, \alpha_M]$, $I_j \subset I_{j-1}$ and $\Gamma \subset I_j$. (For additional details, see Murray and Wright, 1976.) The first point α_ϕ that lies in Γ is taken as α.

In practice, we find that a close approximation to the minimum of $F(x+\alpha p)$ can be obtained after very few estimates α_ϕ (typically 1, 2 or 3). However, the minimum is usually *very close to* α_M. Thus, if an accurate search is performed, at least one variable will become very near to its bound. Sometimes this may be beneficial, but in phase 1 particularly, the danger exists that the optimal value of that variable could be well away from its bound. Although convergence is still assured, the rate of convergence may temporarily be poor.

To guard against this, we set $\beta = 0.999$ and use $0.9\alpha_M$ as an initial step, which is normally accepted. If necessary, we compute the sequence of estimates α_ϕ as described.

4.5. Choice of the barrier parameter. In a "classical" barrier-function method (e.g., as described in Fiacco and McCormick, 1968), the usual procedure is to choose an initial value of μ,

solve the subproblem (2.2), and then decrease μ (say, by multiplying by a constant). In order for $x^*(\mu)$ to converge to x^*, it is essential that $\mu \to 0$. If the barrier search direction and a steplength as defined in Section 4.4 are used to solve (2.2) with a fixed μ, standard proofs for descent methods (see, e.g., Ortega and Rheinboldt, 1970) can be applied to guarantee convergence to $x^*(\mu)$. When (2.1) is non-degenerate and μ is "sufficiently small" (say, $\mu = \mu_{\min}$) it follows from (2.3*a*) that the final iterate of the barrier method will approximate the solution of the linear program (2.1) to within the accuracy specified by $\mu_{\min}$ for a non-degenerate problem. If the problem is degenerate, (2.3*b*) implies that the solution will be less accurate.

Various strategies for changing μ can be devised. The main aim is to reduce μ as quickly as possible, subject to ensuring steady progress toward the solution. For example, only a *single step* of Newton's method could be performed for each of a decreasing sequence of μ-values. Alternatively, each value of μ could be retained until the new iterate satisfies some convergence criterion for the subproblem.

The vector $r = D\eta - \mu e = D(g - A^T\pi)$ may be used to measure convergence for the current subproblem, since it is a scaled form of $g - A^T\pi$ (the reduced gradient for the subproblem), which must tend to zero for any fixed μ. The size of $\|r\|$ is monitored in our implementation, and the reduction of μ is controlled by two parameters as follows.

1. An initial "target level" for $\|r\|$ is defined to be $\tau = \|r_0\| *$ **RGFAC**.
2. Whenever $\|r\| \leq \tau$, the barrier parameter is reduced to $\mu *$ **MUFAC**, τ is recomputed, and a new target level is defined to be $\tau = \|r\| *$ **RGFAC**.

The parameters **RGFAC** and **MUFAC** should lie in the range $(0, 1)$ to be meaningful. For example, the values **RGFAC** = 0.99, **MUFAC** = 0.25 allow a moderate reduction in μ almost every iteration, while **RGFAC** = **MUFAC** = 0.001 requests more discernible progress towards optimality for each subproblem, with a substantial reduction in μ on rare occasions.

4.6. Convergence tests. Three other parameters are needed to define the initial and final values of the barrier parameter, and the degree of optimality required for the final subproblem. In an effort to allow for poor scaling in the data, we initialize μ to

$$\mu_0(1 + |c^T x_0|)/(n \ln(1 + \|x_0\|))$$

for some μ_0, and whenever a newly reduced μ is smaller than

$$\mu_{\min}(1 + |c^T x|)/(n \ln(1 + \|x\|))$$

for the current x, μ is fixed at the latter value for the remaining iterations. At the same time, the target level for the reduced gradient is fixed at

$$\tau_{\min}\|x\|\|\pi\|/n$$

and termination occurs when $\|r\|$ subsequently falls below that value.

In our experiments we have used $\mu_0 = 0.1$, $\mu_{\min} = 10^{-7}$, and $\tau_{\min} = 10^{-7}$. If μ_0 is too large a danger exists on problems for which the feasible region $Ax = b$, $x \geq 0$ is unbounded; since the barrier function is then unbounded below, the iterates can diverge in phase 1 before the artificial variable ξ reaches zero.

5. Numerical Results

5.1. Performance of the barrier method on a standard test set. In this section we summarize the performance of the barrier algorithm described in Section 4 on problems from an LP test set in use at the Systems Optimization Laboratory. All problems are in the form (2.1). To obtain constraints of the form $Ax = b$, any general inequality constraints are converted to equalities using slack variables. Details of the problems are summarized in Table 1. The value of "rows" refers to the number of general constraints, and "columns" to the number of variables, excluding slacks. The number "slacks" is defined above. The column "A" gives the number of nonzeros in the problem. This figure includes one for each slack but excludes the nonzeros in b and c.

The runs summarized in Tables 2–6 were made in double precision on an IBM 3081K (relative precision 2.2×10^{-16}). The source code was compiled with the IBM Fortran 77 compiler VS Fortran, using `NOSDUMP`, `NOSYM` and `OPT(3)`.

Table 2 gives the number of iterations and CPU-seconds required by the the primal simplex method, as implemented in the Fortran code *MINOS 5.0* (May 1985). The default values of the parameters were used throughout (see Murtagh and Saunders, 1983). Results are also given in the case where the constraints are scaled by an iterative procedure that makes the matrix coefficients as close as possible to one.

Many runs were made incorporating different choices for the parameters `RGFAC` and `MUFAC`, which specify the accuracy of a given subproblem and the rate at which the barrier parameter is reduced. One aim was to find a set of values that could be used reliably on all problems. It was found that `RGFAC` = 0.1 and `MUFAC` = 0.1 gave the most consistent results. Table 3 summarizes the performance of the barrier method with these values. The second and third columns of the table give the number of iterations to obtain a feasible point and the total iterations required to satisfy the convergence tests of Section 4.6. The fourth column gives the total CPU time (in seconds) to solve the problem. The values of the optimal objective function found by *MINOS 5.0* were used to judge the accuracy of the final objective in the barrier runs. The underlined digits in the fifth column show the correct figures in the objective function on termination. The final two columns indicate the degree of feasibility and optimality of the final point.

Table 4 gives the results of applying the barrier algorithm with the same scaling procedure as in *MINOS 5.0*. Note that scaling alters the starting point $\|b\|e$, but otherwise its effect was substantial only on *Share1b*.

Table 5 illustrates the performance of a single-phase method in which a composite objective function of the form $\omega c^T x + \xi$ was used throughout (see Section 4.2). The number of phase 1

iterations is sometimes greater than that for $\omega = 0$ (*cf.* Table 4), but the total number of iterations is generally less.

Some statistics concerning the matrix factorizations used in *MINOS 5.0* and the barrier method are provided in Table 6. As in Table 1, column "A" gives the number of nonzeros in the problem. The columns "B" and "$L + U$" give the number of nonzeros in the simplex basis and its LU factors after the last refactorization (this is typically the most dense factorization of all those performed). Finally, the column "R" contains the number of nonzeros in the Cholesky factorization (4.5) required by the barrier method.

Table 1
Problem Statistics

Problem	Rows	Slacks	Columns	A	$\|x^*\|$	$\|\pi^*\|$
Afiro	27	19	32	102	$9.7\,10^2$	$3.9\,10^1$
ADLittle	56	41	97	424	$6.1\,10^2$	$6.2\,10^2$
Share2b	96	83	79	777	$1.8\,10^2$	$3.8\,10^2$
Share1b	117	28	225	1179	$1.3\,10^6$	$7.7\,10^1$
Beaconfd	173	33	262	3408	$1.6\,10^5$	$1.2\,10^2$
Israel	174	174	142	2443	$9.1\,10^5$	$5.6\,10^2$
BrandY	220	54	249	2202	$6.5\,10^4$	$8.7\,10^1$
E226	223	190	282	2768	$9.6\,10^2$	$4.1\,10^1$
BandM	305	0	472	2494	$1.5\,10^3$	$3.0\,10^1$

Table 2
Results from the primal simplex code *MINOS 5.0*

	Optimal	No scaling			With scaling		
	Objective	Phase 1	Total	Time	Phase 1	Total	Time
Afiro	−4.6475314	2	6	0.5	2	6	0.5
ADLittle	225494.96	28	123	1.3	30	98	1.1
Share2b	−415.73224	59	91	1.3	74	121	1.4
Share1b	−76589.319	135	296	3.4	144	260	2.8
Beaconfd	33592.486	8	38	1.9	6	39	1.8
Israel	−896644.82	109	345	5.0	41	231	3.7
BrandY	1518.5099	176	292	4.9	216	377	5.9
E226	−18.751929	109	570	9.4	101	471	7.5
BandM	−158.62802	167	362	7.6	280	534	10.0

Table 3

No scaling, RGFAC = 0.1, MUFAC = 0.1

Problem	Phase 1	Total	Time	Objective	$\frac{\|b-Ax\|}{\|b\|}$	$\frac{\|D(c-A^T\pi)\|}{\|x\|\|\pi\|}$
Afiro	4	20	0.4	−464.75314	$3.6\,10^{-12}$	$1.9\,10^{-9}$
ADLittle	13	36	1.1	225494.96	$8.8\,10^{-10}$	$2.4\,10^{-10}$
Share2b	7	22	1.5	−415.73224	$2.1\,10^{-8}$	$9.3\,10^{-11}$
Share1b	11	66	4.9	−76589.319	$9.8\,10^{-8}$	$4.5\,10^{-11}$
Beaconfd	20	40	9.9	33592.486	$8.4\,10^{-9}$	$4.5\,10^{-11}$
Israel	17	54	22.6	−896644.82	$8.6\,10^{-6}$	$9.4\,10^{-12}$
BrandY	19	40	8.5	1518.5099	$3.8\,10^{-8}$	$3.7\,10^{-11}$
E226	18	45	9.8	−18.751929	$8.0\,10^{-8}$	$1.4\,10^{-10}$
BandM	19	41	9.3	−158.62802	$3.7\,10^{-8}$	$5.6\,10^{-11}$

Table 4

With scaling, RGFAC = 0.1, MUFAC = 0.1

Problem	Phase 1	Total	Time	Objective	$\frac{\|b-Ax\|}{\|b\|}$	$\frac{\|D(c-A^T\pi)\|}{\|x\|\|\pi\|}$
Afiro	4	20	0.4	−464.75314	$6.3\,10^{-12}$	$9.6\,10^{-10}$
ADLittle	13	34	1.1	225494.96	$6.0\,10^{-10}$	$2.2\,10^{-10}$
Share2b	8	24	1.6	−415.73224	$4.9\,10^{-9}$	$5.4\,10^{-11}$
Share1b	7	33	2.8	−76589.319	$4.7\,10^{-9}$	$4.7\,10^{-12}$
Beaconfd	22	42	11.1	33592.486	$1.9\,10^{-7}$	$1.5\,10^{-10}$
Israel	11	58	24.1	−896644.82	$1.6\,10^{-8}$	$5.0\,10^{-12}$
BrandY	18	42	9.2	1518.5099	$1.3\,10^{-7}$	$1.8\,10^{-11}$
E226	18	46	10.2	−18.751929	$1.1\,10^{-7}$	$1.4\,10^{-11}$
BandM	20	43	9.8	−158.62802	$1.7\,10^{-8}$	$4.3\,10^{-11}$

Table 5

Composite objective function

With scaling, RGFAC = 0.1, MUFAC = 0.1, $\omega = 0.01/\|e\|$

Problem	Phase 1	Total	Time	Objective	$\frac{\|b-Ax\|}{\|b\|}$	$\frac{\|D(c-A^Ty)\|}{\|x\|\|c\|}$
Afiro	4	20	0.4	−464.75314	$5.5\,10^{-12}$	$9.3\,10^{-10}$
ADLittle	17	31	1.0	225494.96	$6.3\,10^{-8}$	$9.7\,10^{-11}$
Share2b	8	24	1.6	−415.73224	$4.4\,10^{-10}$	$5.4\,10^{-11}$
Share1b	7	33	2.8	−76589.319	$4.4\,10^{-9}$	$4.6\,10^{-12}$
Beaconfd	24	32	9.6	33592.486	$2.2\,10^{-7}$	$2.5\,10^{-11}$
Israel	11	62	25.1	−896644.82	$6.2\,10^{-9}$	$4.8\,10^{-11}$
BrandY	25	35	8.1	1518.5099	$6.7\,10^{-7}$	$9.8\,10^{-13}$
E226	21	40	9.4	−18.751929	$6.5\,10^{-8}$	$1.5\,10^{-11}$
BandM	23	35	8.9	−158.62802	$1.4\,10^{-6}$	$3.2\,10^{-11}$

Table 6

Factorization Statistics

Problem	A	B	$L+U$	R
Afiro	102	67	67	80
ADLittle	424	261	275	355
Share2b	777	564	597	925
Share1b	1179	579	636	1345
Beaconfd	3408	1546	1546	2727
Israel	2443	1644	1664	3533†
BrandY	2202	1318	1485	3251
E226	2768	1440	1620	3416
BandM	2494	2016	2372	4355

†11259 if six dense columns are included.

5.2. Obtaining an optimal basic solution. By its very nature, a barrier method can at best terminate somewhere "close" to an optimum. We must then ask: how close is "close", and which of the several characterizations of LP optimality are we close to achieving?

In practice, LP users (and their report-writing programs) expect alleged optimal solutions to be both primal and dual feasible, thus exhibiting complementary slackness. The last column in Tables 3–5 show that the barrier algorithm can attain complementary slackness to high precision. However, LP users also expect their solutions to be basic. This can be achieved by taking the final solution from the barrier algorithm and processing it through the *BASIC* procedure common

to most mathematical programming systems.

The *BASIC* procedure (sometimes known as *INSERT by value*; see Benichou *et al.*, 1977) takes a set of variable names and values and produces a basic solution that has at least as good an objective value or sum of infeasibilities. The simplex method may then be applied to reach optimality. The time required by *BASIC* and the simplex method together, compared to the time required by the simplex method alone, provides another practical measure of closeness to optimality.

Some experiments of this kind were performed to test the quality of the solutions obtained by the barrier algorithm. An early implementation of the barrier algorithm was used, with a slightly different starting point for phase 1 and a different strategy for controlling μ. This strategy initialized μ to $10\|c\|/n$ and multiplied μ by 0.25 every iteration if μ was still larger than 10^{-8}. The algorithm was terminated when $\max_j |x_j \eta_j| \leq 10^{-7}\|c\|\|x\|$, i.e., when complementarity was approximately achieved.

These experiments were carried out on an IBM 3033N, except for one problem *Degen3* that was run on an IBM 3081K. The problems used were an available subset of those in Table 1, plus a graduated set of three models from a single application, ranging from small to medium in size (see Table 7) and notable for their severe degeneracy.

Initially, all problems were solved from scratch by Ketron's *WHIZARD* optimizer, which was called from MPSIII in all cases except for *Degen3*, where the host was MPSX/370. The first three columns of Table 8 give the number of columns selected for the initial basis by *CRASH*, the number of simplex iterations required to reach optimality, and the CPU time in seconds. (All times in this section are for a complete MVS job step, including model input and solution output.)

The next three columns of Table 8 give results for the barrier algorithm. For the smaller problems, there is about 10 percent overhead for input, output, symbolic ordering (subroutine *Genqmd*) and symbolic factorization (subroutine *Smbfct*). For the larger problems, the cost of the single call to *Genqmd* became significant: 2.4 seconds for *BandM*, 13.4 seconds for *Degen2*, and 237 seconds for *Degen3*. Clearly some more efficient means of preprocessing must be found for large problems.

Table 7 compares the number of nonzeros in a typical *WHIZARD* basis factorization with the number of nonzeros in the Cholesky factors R, again indicating the high cost of solving large least-squares problems.

The last two columns of Table 8 show the work required by *BASIC* and the simplex method to reach optimality, starting from the point obtained by the barrier algorithm. With $\hat{x}$ denoting the basic solution obtained by *BASIC*, the quantities $\|b - A\hat{x}\|/\|b\|$ were observed to be less than 10^{-5} in all cases, and the values of $|c^T\hat{x} - c^Tx^*|/|c^Tx^*|$ were all less than 10^{-3}. The number of subsequent simplex iterations appears to be a function of size and degeneracy, the two being closely related in this sample. Clearly, the primary effect of the post-*BASIC* iterations is to remove dual infeasibilities.

The problems *BrandY*, *E226* and *BandM* have 23, 20 and 21 structurally degenerate variables respectively. Ideally, these should be removed by the *PRESOLVE* procedure before entering the barrier (or simplex) method. They can be restored and dual feasibility attained at trivial cost by the *POSTSOLVE* procedure (see Tomlin and Welch, 1983). At present there is no obvious way of circumventing extreme degeneracy of the ordinary type, such as that displayed by the *Degen* problems. However, the relative number of post-*BASIC* simplex iterations required for these problems appears to decline with problem size, compared to the number required when starting from scratch. It may well be that non-simplex methods such as the projective and barrier methods will prove most valuable for dealing with very degenerate LPs.

Table 7

Model statistics — degenerate problem set

Problem	Rows	Slacks	Columns	A	B	$L+U$	R
Degen1	66	15	72	296	249	251	514
Degen2	444	223	534	4894	3076	3718	16243
Degen3	1503	786	1818	25432	18468	20322	119373

Table 8

Results from the *BASIC* procedure

	Whizard			Barrier			*BASIC*	
	Crash	Simplex	Time	Phase 1	Total	Time	Simplex	Time
Afiro	18	4	0.5	3	14	0.3	0	0.2
ADLittle	18	80	1.0	8	23	1.2	1	0.3
Share2b	13	84	1.1	6	17	1.8	2	0.4
BrandY	85	168	2.7	12	29	10.8	6	2.3
E226	50	350	3.9	15	29	13.0	6	2.2
BandM	114	253	3.5	15	28	13.0	5	2.8
Degen1	38	23	0.7	2	15	0.9	18	0.7
Degen2	187	2650	31.2	13	26	54.9	300	7.3
Degen3	590	8889	226.0	11	25	528.0	593	60.0

6. Future Developments and Conclusions

6.1. Solving the least-squares subproblem. The present implementation, as in Gay (1985), uses a preconditioned conjugate-gradient method to solve the relevant least-squares subproblems. This approach allows the use of existing software for computing Cholesky factors, and provides a

convenient way of dealing with a few dense columns of A that would degrade the sparsity of those factors. Perhaps further efficiency could be gained by discarding small nonzeros in the product AD^2A^T (not just rows and columns of A) before computing its Cholesky factors. However, a new symbolic factorization would then be required at every stage, not just once.

Unfortunately, it seems that the quality of the preconditioner must remain rather high throughout, since an iteration of *LSQR* or the conjugate-gradient method requires two matrix-vector products involving A and two solves with the preconditioner R, and is therefore as expensive (typically) as two iterations of the simplex method. If the average number of iterations required by *LSQR* were 20, say, then the barrier algorithm would have to terminate in about 1/50th the number of iterations in order to be competitive with the simplex method, *even if minimal effort were required to obtain R each iteration.*

To illustrate, the test problem *Israel* has six dense columns in A. With these excluded from the computation of R, *LSQR* required an average of 12 iterations, and the run time was correspondingly high. (On the other hand, retaining all columns gave an R with three times as many nonzeros and a run-time twice as great.)

For reasons such as these, we remain doubtful that good preconditioners can be computed with adequate efficiency for *arbitrary* matrices DA^T, i.e., for arbitrary linear programs. Only for special classes of problem does there seem to be room for optimism; for example, those exhibiting a block-triangular structure with many *small* diagonal blocks.

In place of the iterative methods just described, one can employ a sparse orthogonal factorization of the form

$$DA^T = Q\begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad Q^TQ = I, \qquad R \text{ upper triangular} \tag{6.1}$$

to solve the least-squares problems directly, where R is analytically the same as the Cholesky factor of AD^2A^T. General-purpose software exists for this computation, in particular *SPARSPAK-B* (George and Ng, 1984), which has excellent numerical properties and is able to treat dense rows of DA^T specially in order to preserve the sparsity of R. Its use in this context merits future investigation.

A further direct approach is to apply a sparse indefinite solver to the symmetric system (2.10). The *MA27* package of Duff and Reid (1982) is applicable, and we believe it shows considerable promise. As with the sparse QR (6.1), a single symbolic factorization would serve all iterations. In addition, dense columns in A would not drastically affect the sparsity of the factors.

6.2. Adjusting the barrier parameter. Numerous authors have suggested extrapolation techniques in connection with barrier functions. We have conducted some limited experiments, as follows. The barrier function was minimized reasonably accurately for two quite large values of μ ($\|c\|/n$ and $0.1\|c\|/n$), and extrapolation was then performed. The resulting solution was accurate to about five figures. However, it is difficult to evaluate the practical merits of this

approach without further study. Recall that such a strategy would need to be applied to both phases.

A number of suggestions have been made for automating the choice of μ. The method of centers (see Fiacco and McCormick, 1968; Huard, 1967) is in essence a barrier-function method in which a transformation is also applied to the objective function, thereby negating the need for a barrier parameter. See also Todd and Burrell (1985) for a discussion of this approach.

It should be stressed, however, that the freedom to choose μ may well be an asset, especially in the case of linear programs. This is confirmed by our experience with the conservative strategy of allowing μ to be reduced only occasionally. Considerable progress is then often achieved before the least-squares problems become unduly ill-conditioned.

6.3. Use of the entropy function. Because of the similarities, we note the work of many authors on incorporating the entropy function into linear programming models. In place of subproblem (2.2), one can consider the subproblem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \mu \sum_{j=1}^{n} x_j \ln x_j \\ \text{subject to} \quad & Ax = b, \end{aligned}$$

where the scalar μ ($\mu > 0$) is again specified for each subproblem. Erlander (1977) reviews problems of this kind and suggests Newton-type methods for their solution. Computational algorithms have been developed by Eriksson (1980, 1981).

If a feasible-point descent method is applied as in Section 2, the Newton search direction and Lagrange-multiplier estimates satisfy the system

$$\begin{pmatrix} \mu D^{-1} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -p \\ \pi \end{pmatrix} = \begin{pmatrix} c + \mu v \\ 0 \end{pmatrix}$$

in place of (2.9), where $D = \operatorname{diag}(x_j)$ and v has components $v_j = 1 + \ln x_j$. A least-squares subproblem follows as before. In terms of the algorithm of Section 4.1, r would be defined as $r = D^{1/2}(\eta + \mu v)$ in steps 1, 3 and 5, and D would be changed to $D^{1/2}$ in the least-squares problem (4.1). Finally, we would have $p = -(1/\mu)D^{1/2}r$ in step 5.

The entropy function is convex and (unlike the logarithmic barrier function) it is bounded below. Also, since its Hessian is μD^{-1} rather than μD^{-2}, the least-squares problems should be better conditioned as the LP solution is approached. Further computational work therefore seems to be justified, either as in Eriksson (1980, 1981) or along the lines suggested here.

6.4. Conclusions. Most qualitative aspects of Karmarkar's projective method can be found in the projected Newton barrier algorithm described here. The nonlinear programming viewpoint has provided us with an armory of known theoretical and practical techniques, to be applied to convergence analysis and computer implementation. To date, a casualty appears to be the proof of polynomial complexity. While this may seem a serious loss, the number of iterations

required by the barrier algorithm appears to be similar in practice to that reported for various projective methods (*cf.* Tomlin, 1985, and Lustig, 1985), and our implementation has proved to be competitive with the simplex method on *some* of a limited class of moderate-sized problems (all of which are degenerate and poorly scaled).

Two facts remain:

- large-scale least-squares problems can be very expensive to solve (compared to square systems $Bx = b$);
- interior-point methods are inescapably minimizing a highly nonlinear function.

In view of the second point, poor performance can be expected if variables migrate prematurely towards the singularities at their bounds. It has been difficult to develop a robust algorithm for this reason. The same difficulty arises if a "good" starting point is available from an earlier run on a similar problem (by far the most common situation), since such a solution will have many variables on or close to their bounds.

From the computational evidence to date, the future for interior-point methods seems brightest in the context of cold-start solution of very large, specially structured problems.

Acknowledgements

The authors would like to thank Ketron, Inc., for providing machine time to carry out some of the computational experiments. Our thanks go also to George Dantzig for his encouragement and bibliographical assistance, and to Irvin Lustig for his helpful and timely comments.

References

Benichou, M., Gauthier, J. M., Hentges, G. and Ribière, G. (1977). The efficient solution of large-scale linear programming problems — some algorithmic techniques and computational results, *Math. Prog.* **13**, pp. 280–322.

Brent, R. P. (1973). *Algorithms for Minimization without Derivatives*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Dantzig, G. B. (1963). *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey.

Duff, I. S. and Reid, J. K. (1982). *MA27* — a set of Fortran subroutines for solving sparse symmetric sets of linear equations, Report AERE R-10533, Computer Science and Systems Division, AERE Harwell, Oxfordshire, England.

Eriksson, J. (1980). A note on solution of large sparse maximum entropy problems with linear equality constraints, *Math. Prog.* **18**, pp. 146–154.

Eriksson, J. (1981). *Algorithms for entropy and mathematical programming*, Ph. D. Thesis, Department of Mathematics, Linköping University, Linköping, Sweden.

Erlander, S. (1977). Entropy in linear programs — an approach to planning, Report LiTH-MAT-R-77-3, Department of Mathematics, Linköping University, Linköping, Sweden.

Fiacco, A. V. and McCormick, G. P. (1968). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York and Toronto.

Fletcher, R. (1981). *Practical Methods of Optimization, Volume 2*, John Wiley and Sons, Chichester and New York.

Fletcher, R. and McCann, A. P. (1969). "Acceleration techniques for nonlinear programming", in *Optimization* (R. Fletcher, ed.), pp. 203–213, Academic Press, London and New York.

Frisch, K. R. (1955). The logarithmic potential method of convex programming, Memorandum of May 13, 1955, University Institute of Economics, Oslo, Norway.

Frisch, K. R. (1957). Linear dependencies and a mechanized form of the multiplex method for linear programming, Memorandum of September, 1957, University Institute of Economics, Oslo, Norway.

Gay, D. M. (1985). Informal presentation on solving sparse least-squares problems, February 28, Department of Operations Research, Stanford University, California.

George, J. A. and Liu, J. W. (1981). *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ.

George, J. A. and Ng, E. (1984). A new release of *SPARSPAK* — the Waterloo sparse matrix package, *SIGNUM Newsletter* **19**, 4, pp. 9–13.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press, London and New York.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1982). Sparse matrix methods in optimization, *SIAM Journal on Scientific and Statistical Computing* **5**, pp. 562–589.

Heath, M. T. (1984). Numerical methods for large sparse linear least squares problems, *SIAM Journal on Scientific and Statistical Computing* **5**, pp. 497–513.

Hoffman, A. J., Mannos, M., Sokolowsky, D. and Wiegmann, N. (1953). Computational experience in solving linear programs, *Journal of the Society for Industrial and Applied Mathematics* **1**, pp. 17–33.

Huard, P. (1967). Resolution of mathematical programming with nonlinear constraints by the method of centres, in *Nonlinear Programming* (J. Abadie, ed.), pp. 207–219, North-Holland, Amsterdam.

Jittorntrum, K. (1978). *Sequential Algorithms in Nonlinear Programming*, Ph. D. Thesis, Australian National University.

Jittorntrum, K. and Osborne, M. R. (1978). Trajectory analysis and extrapolation in barrier function methods, *Journal of Australian Mathematical Society* **21**, pp. 1–18.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming, *Proceedings of the 16th Annual ACM Symposium on the Theory of Computing*, pp. 302–311.

Khachiyan, L. G. (1979). A polynomial algorithm in linear programming, *Doklady Akademiia Nauk SSSR Novaia Seriia* **244**, pp. 1093–1096. [English translation in *Soviet Mathematics Doklady* **20**, (1979), pp. 191–194.]

Lustig, I. J. (1985). A practical approach to Karmarkar's algorithm, Report SOL 85-5, Department of Operations Research, Stanford University, California.

Mifflin, R. (1972a). Convergence bounds for nonlinear programming algorithms, Administrative Sciences Report 57, Yale University, Connecticut.

Mifflin, R. (1972b). "On the convergence of the logarithmic barrier function method", in *Numerical Methods for Non-Linear Optimization* (F. Lootsma, ed.), pp. 367–369, Academic Press, London and New York.

Murray, W. and Wright, M. H. (1976). Efficient linear search algorithms for the logarithmic barrier function, Report SOL 76-18, Department of Operations Research, Stanford University, California.

Murtagh, B. A. and Saunders, M. A. (1983). *MINOS 5.0* user's guide, Report SOL 83-20, Department of Operations Research, Stanford University, California.

Ortega, J. M. and Rheinboldt, W. C. (1970). *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, London and New York.

Paige, C. C. and Saunders, M. A. (1982). *LSQR*: An algorithm for sparse linear equations and sparse least-squares, *ACM Transactions on Mathematical Software* **8**, pp. 43–71.

Todd, M. J. and Burrell, B. P. (1985). An extension of Karmarkar's algorithm for linear programming using dual variables, Report 648, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York.

Tomlin, J. A. (1985). An experimental approach to Karmarkar's projective method for linear programming, Manuscript, Ketron Inc., Mountain View, California.

Tomlin, J. A. and Welch, J. S. (1983). Formal optimization of some reduced linear programming problems, *Math. Prog.* **27**, pp. 232–240.

Tompkins, C. B. (1955). "Projection methods in calculation", in *Proceedings of the Second Symposium in Linear Programming* (H. A. Antosiewicz, ed.), pp. 425–448, United States Air Force, Washington, D. C.

Tompkins, C. B. (1957). Some methods of computational attack on programming problems, other than the simplex method, *Naval Research Logistics Quarterly* **4**, pp. 95–96.

Vanderbei, R. J., Meketon, M. S. and Freedman, B. A. (1985). A modification of Karmarkar's linear programming algorithm, Manuscript, AT&T Bell Laboratories, Holmdel, New Jersey.

von Neumann, J. (1947). "On a maximization problem", Manuscript, Institute for Advanced Study, Princeton University, Princeton, New Jersey.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: SOL 85-11	2. GOVT ACCESSION NO.: AD-A158 212
	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Philip E. Gill, Walter Murray, Michael A. Saunders, J.A. Tomlin, Margaret H. Wright	8. CONTRACT OR GRANT NUMBER(s): N00014-85-K-0343 DAAG29-84-K-0156
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Operations Research - SOL Stanford University Stanford, CA 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR-047-064
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217	12. REPORT DATE: July 1985
	13. NUMBER OF PAGES: 23 pp.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

linear programming, interior-point methods, logarithmic barrier function, Karmarkar's projective method

20. ABSTRACT (Continue on reverse side if necessary and identify by block number) We discuss interior-point methods for linear programming derived by applying a logarithmic barrier transformation and performing projected Newton steps for a sequence of barrier parameters. Under certain conditions, one of these "projected Newton barrier" methods is shown to be equivalent to Karmarkar's (1984) projective method for linear programming.

Details are given of a specific barrier algorithm and its practical implementation. Numerical results are given for several nontrivial test problems.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

END

FILMED

9-85

DTIC